(1)

Ultra-Dependable Real-Time Computing

SFRC. N0001491J1304 N173

YEARLY REPORT

1 October 1990 - 30 September 1991

Prepared for:

Department of the Navy
Office of Naval Research
800 North Quincy Street
Arlington, Virginia 22217-5000

Prepared By:

Jay K. Strosnider & Ron Bianchini, Principal Investigators
Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
(412) 268-8725

Principal Investigator Names:
Jay K. Strosnider (412) 268-6927 strosnider@ece.cmu.edu
Ron Bianchini (412) 268-7105 bianchini@ece.cmu.edu
PI Institution: Carnegie Mellon University
Contract title: Ultra-Dependable Real-Time Computing
Contract Number: N0001491J1304 N173
Reporting Period: 1 Oct 90 - 30 Sep 91

1 Productivity Measures

- Refereed papers submitted but not yet accepted: 5
- Refereed papers accepted and in press: 3
- Refereed papers published: 7
- Books submitted or published: none
- Other reports: 2
- Ph.D. dissertations: 1
- Patents filed or granted: none
- Invited presentations: 10
- Contributed presentations: 7
- Honors, Prizes and Awards received:
- Graduate students supported: 2
- Post-docs supported: 0
- Minorities supported: 0

Statement A per telecon James Smith
ONR/Code 1267
Arlington, VA 22217-5000

NWW 4/1/92

Principal Investigator Names:
Jay K. Strosnider (412) 268-6927 strosnider@ece.cmu.edu
Ron Bianchini (412) 268-7105 bianchini@ece.cmu.edu
PI Institution: Carnegie Mellon University
Contract title: Ultra-Dependable Real-Time Computing
Contract Number: N0001491J1304 N173
Reporting Period: 1 Oct 90 - 30 Sep 91

2 Summary of Technical Progress

2.1 Overview

The research effort this year focused in four major areas.

1. Developing schedulable fault recovery strategies that guarantee minimum interference with fault free tasks, and graceful degradation;
2. Developing analytical techniques that support quantitative analysis of computer architectures for fault-tolerant computation;
3. Investigating how intelligent integrated agents can enhance distributed systems' dependability.
4. Developing network diagnostic strategies with provable timing properties.

The project members are also concerned with the transition of the theory being developed to the user community, especially the U.S. Navy. For this reason, project members have a close relationship with the RTSIA (Real-Time Scheduling in Ada) project in the CMU Software Engineering Institute, and Distributed Combat Control Project at Naval Ocean Systems Center in San Diego.

During the October 1, 1989 - September 30, 1990 period, substantial progress was made in each of the two areas listed above. That progress is briefly discussed below.

2.2 Schedulable Fault Recovery

Our research efforts in the past year can be categorized into two broad areas. First, we have continued to make progress in the development of real-time scheduling theoretic policies for handling the non-deterministic service requirements of recovery operations. And second, we have stepped up our efforts to evaluate the potential dependability gains associated with the integration of our proposed time-based recovery schemes into conventional Fault-Tolerant Real-Time (FT/RT) architectures. A brief summary of our main contributions to each area follows.

As a result of an extensive literature survey of practical fault recovery mechanisms, we developed a taxonomy of system-level recovery mechanisms based on time redundancy. Two main time-based recovery mechanisms are distinguished: those which *pre-allocate* time for recovery and those in which time is *dynamically* allocated. Most FT/RT architectures that support time-based recovery do so by following some pre-allocation policy, which is usually **ad-hoc** in nature. In our efforts to develop a general framework for time-based recovery, we formalized the notion of pre-allocation policies and provided insight into their implementation tradeoffs. Furthermore, we developed two dynamic allocation policies for fixed-priority preemptive systems, referred to as the **Transient Server** and the **Dynamic**

Slack Manager. Each of these policies are based on robust scheduling theoretic principles and have formally extended the classical notion of guaranteed task-level timing correctness in the absence of failures to a notion of scheduling stability during recovery. In order to evaluate the relative performance of pre-allocation versus dynamic allocation schemes, we have developed a discrete event simulator called DERTsim (DEpendable Real-Time simulation model). We have begun conducting a series of simulation experiments in which recoverability profiles (i.e., a measure of the ability of the scheduler to accommodate recovery operations without disrupting the timing correctness of the RT application tasks) are generated for a wide range of RT workloads. Our objective is to characterize the expected performance of these different time-based recovery schemes under a variety of loading conditions.

In order to evaluate the potential dependability gains attainable through time-based recovery, we have conducted research into the state-of-art validation tools developed within the Fault-Tolerance community. We have been successful in identifying tools to support three suitable validation methods, namely analytical modeling tools (e.g., HARP, SHARPE), system-level simulation tools (e.g., DEPEND), and fault-injection testbeds (e.g., FAUST, FIAT). We are currently conducting feasibility studies to identify which tool is the most suitable to fulfill our evaluation needs.

2.3 Analysis of Architectures for Fault-Tolerant Computation

This effort developed computational models to examine the relative viability of traditional von Neumann, Very Long Instruction Word (VLIW), and dataflow architectures for fault-tolerant applications requiring modular redundancy. Analytic machine execution cycle equations are developed for each class of machine with respect to varying amounts of voting overhead and granularity. Attributes that are varied for each architecture include shared memory access, program decomposition, uni-tasking and multi-programmed scheduling of the application's algorithm. The models and methodology developed in this research provide a framework for quantitative analysis of very different architectures when executing fault-tolerant voting applications. This work is largely complete with no addition planned effort in the coming year.

2.4 Using Intelligent Integrated Agents to Enhance Distributed System Dependability

This effort has just been kicked off this year and leverages heavily off of the CROPS5 integrated, real-time production system developed earlier. The effort is largely a system engineering effort focusing combining technologies to maximize large system dependability.

2.5 Time-Constrained Diagnostics

A critical issue for distributed real-time applications, including embedded systems, is fault tolerance. This research effort proposes the first application of the large body of theoretical results in distributed system level diagnosis to real-time distributed systems. A distributed self-diagnosis algorithm, EVENT_SELF, has been developed and utilized in the Distributed Self-Diagnosis (DSD) system that combines the rigor associated with previous theoretical results and the resource limitations associated with actual systems. Resource limitations identified in real systems include available message capacity for the communication network and limited processor execution speed. The EVENT_SELF algorithm differs from previously published algorithms by adopting an event-driven approach to self-diagnosis. Algorithm messages are reduced to those messages that are required to indicate changes in system state. Preliminary results have proven the resource overhead required by the EVENT_SELF algorithm to be within reasonable bounds, permitting expansion to large wide area networks.

The resource limitations and predictability requirements of real-time systems place greater restrictions on algorithm execution. Ideally, a real-time distributed self-diagnosis system must impose predictable and minimum utilization of processor and network resources. Predictable processor computation time is required for execution of the distributed diagnosis algorithm and for processor testing. The primary focus of this work is to evaluate the distributed diagnosis techniques within the bounds of real-time systems constraints. Additional theoretical work will address the fault coverage of the system. This includes the ability to handle intermittent or hybrid fault conditions as shown in and the examination of improved processor testing schemes to identify a higher percentage of processor faults.

Results of this research project for the first year focus on the specification and implementation of a real-time version of the DSD algorithm, called RT-DSD. The RT-DSD task set has been defined and validated to perform the diagnosis task. A preliminary version of the algorithm, that does not include the event-driven enhancement, has been implemented. Current research emphasis is placed on the implementation of the final version that will include the event-driven enhancement. Once implemented, the task set will be analyzed and evaluated under experimentation.

2.5.1 Scheduling

To implement a real-time version of DSD, a fixed priority scheduling algorithm is chosen. Fixed priority scheduling is chosen since it is easy to implement with most operating systems and the analysis procedure for several fixed priority schemes is well documented. The scheduling method employed for the RT-DSD algorithm is deadline monotonic scheduling, i.e. tasks with shorter deadlines have higher priority.

2.5.2 Platform

The Ada language was chosen for implementation of the RT-DSD algorithm. The Verdix Ada Development System (VADS) is currently utilized with a cross compiler for the Motorola 68020. Ada was selected mainly due to its tasking support and its built in fixed priority scheduling. Ada rendezvous are augmented with semaphores provided by the VADS system. The VADS system provides additional support for real-time dynamic memory management.

The testbed chosen to implement the algorithm was four Motorola 68020 nodes, connected through a Real-Time Communication Network (RTCN). To simulate a larger network, each node is programmed to simulate multiple independent nodes. The interconnection network is an experimental fiber-optic LAN designed by IBM. RTCN was created specifically to prevent priority inversion in LAN communications. By meeting this requirement, RTCN can be used to implement real-time scheduling algorithms, such as rate-monotonic scheduling, for communication tasks. Additionally, RTCN provides high level functionality, such as connection and connectionless service, atomic send-with-acknowledge, and DMA capability to off-load communication processing.

2.5.3 Prototype and Final Implementation

A prototype version of RT-DSD was implemented for preliminary evaluation of the platform. The prototype implements the NEW_SELF algorithm as a real-time task set. No consideration is given for event-driven messages. The prototype was completed in June, 1991 and was successful in detecting both node failures and recoveries. An implementation of the EVENT_SELF algorithm for real-time is in progress. The event driven RT-DSD algorithm consists of eight Ada tasks, with separate deadlines and priorities.

2.5.4 Analysis Approach

The goal of the analysis is to determine a real-time task set for RT-DSD, where each task is defined by its execution time, deadline, and period. Each network node or CPU will perform the RT-DSD task set, and a communication task set that is supported by RTCN. The entire task set will be analyzed to determine the total resources to be reserved for RT-DSD to operate with hard diagnosis deadlines. Resource requirements are expected to be functions of network size, diagnosis latency, resource failure rate and the tests performed. The resulting task set will be scheduled by the deadline monotonic real-time scheduling algorithm.

The original DSD algorithm is poorly behaved for real-time systems in terms of predictable task execution time and rate of task arrival, both of which vary with the node failure rate. To specify the resource requirements of the program, the node failure rate is limited to the number of fault events diagnosable per unit time. Any number of fault events can occur, but only up to the limit can be diagnosed in the time unit. The RT-DSD algorithm is further complicated by its nature as a distributed task. The presence of real-time communications, as found in RTCN, is crucial in determining the timing of the overall system diagnosis. Task dependencies, specifically when a task invokes another task, are presently being analyzed to determine the number and type of tasks that will be initiated under a variety of fault conditions. Since the task set generated by RT-DSD is flexible, the worst cast task set will used in subsequent analysis. The worst case task set occurs when the largest allowed number of node failures take place within a single test period.

Dependency analysis can also be used to determine task sequencing, both within and across node boundaries. Once task sequencing is known, the diagnosis latency of the algorithm can be stated as a function of task deadlines. Task execution times will then be measured, under the desired fault conditions. Execution times for software will be fixed for the given experimental platform, but the methodology will be applicable to any particular implementation. A hard deadline is assigned to each task of the task set. The resultant task set is analyzed to ensure real-time schedulability. Task deadlines are used to accurately determine the hard-deadline diagnosis latency of the system. The analysis is iterated for various fault cases and deadline assignments, allowing the diagnosis capability of RT-DSD to be tailored to specific applications.

After analysis, the resources that must be reserved for RT-DSD to guarantee hard-deadline diagnosis will be characterized. This task set can be incorporated into a real-time system by combining all the tasks of RT-DSD with those from a desired application, assigning priorities appropriately, and analyzing the overall task set for schedulability.

2.5.5 Further Considerations: Overload and Degradation

The actual execution time of the RT-DSD algorithm tasks can vary significantly with the number of faults in the system. The CPU and communication resources are reserved only for a subset of faults, specifically for fault sets that occur within the specified failure rate. The RT-DSD algorithm requires more processing time and network bandwidth than specified if the failure rate exceeds its limit, causing other tasks to miss deadlines. To prevent this case, the RT-DSD algorithm should be implemented with a sporadic server, allowing it to execute normally when the failure rate is within design limits, but limiting its execution time during overload transients. Under overload, the RT-DSD algorithm will not meet deadlines, but other tasks in the system will not be affected. Furthermore, the diagnosis latency of the RT-DSD algorithm will degrade, but will converge to correct diagnosis.

Principal Investigator Names:
Jay K. Strosnider (412) 268-6927 strosnider@ece.cmu.edu
Ron Bianchini (412) 268-7105 bianchini@ece.cmu.edu
PI Institution: Carnegie Mellon University
Contract title: Ultra-Dependable Real-Time Computing
Contract Number: N0001491J1304 N173
Reporting Period: 1 Oct 90 - 30 Sep 91

3 Publications and Presentations

The following are project publications which were written, appeared or are in press for the above period. Project members made many research presentations during the contract period, and these are not enumerated. Each of the articles listed below that was published in a conference proceedings volume was presented at that conference.

3.1 Published or In Press

- R.P. Bianchini Jr., and R. Buskens. Implementation of On-line Distributed System-Level Diagnosis Theory, IEEE Transactions on Computers, Special Issue Fault-Tolerant Computing, May 1992.
- S. Thuel and J.K. Strosnider, The Transient Server Approach to Scheduling Time-Critical Recovery Operations. Proceedings of the 12th IEEE Real-Time Systems Symposium, December 1991.
- C.J. Paul, A. Acharya, B. Black and J.K. Strosnider. Reducing Problem-Solving Variance to Improve Predictability, Communications of the ACM, August 1991, pages 80-93.
- L.E. Holloway, C.J. Paul, J.K. Strosnider, and B.H. Krogh, Integration of Behavioral Fault-Detection Models and an Intelligent Reactive Scheduler. Proceedings of Intelligent Controls Conference, Washington, D.C., August 1991.
- R.P. Bianchini Jr., and R. Buskens. An Adaptive Distributed System-Level Diagnosis Algorithm and Its Implementation, The twenty-first Annual International Symposium on Fault-Tolerant Computing, June, 1991.
- C.J. Paul, A. Acharya, B.Black and J.K. Strosnider, Adding Problem Solving Capabilities to Existing Real-Time Systems, Proceedings of the Eighth IEEE Workshop on Real-Time Operating Systems and Software, May 1991.
- R. Mraz, G. Palmer and J.K. Strosnider, Analysis of Architectures for Fault-Tolerant Computation, 24th Hawaii International Conference on System Sciences, Award Paper, January 1991.
- D.B. Kirk and J.K. Strosnider, SMART Cache Design Using the R3000, Proceedings of the 11th IEEE Real-Time Systems Symposium, December 1990.

3.2 Submitted for Publication

- R. Mraz, G. Palmer, J.K. Strosnider, Analysis of Architectures for Fault-Tolerant Computation, IEEE Transactions on Reliability, Submitted February 1991.
- J.E. Sasinowski and J.K. Strosnider, A Dynamic Programming Algorithm for Cache/Memory Partitioning for Real-Time Systems, IEEE Transactions on Computers, Submitted October, 1991.

- S. Sathaye, A. Lin, R.B. Bianchini and J.K. Strosnider, Routing Periodic Real-Time Traffic in a Packet Switched Network, 12th International Conference on Distributed Systems, Yokohama, Japan, June 9-12, 1992 (submitted 10-91)

3.3 Ph.D. Dissertations

- Kirk, David, "Predictable Cache Design for Real-Time Systems," Department of Electrical and Computer Engineering, Carnegie Mellon University,) December, 1990.

Principal Investigator Names:
Jay K. Strosnider (412) 268-6927 strosnider@ece.cmu.edu
Ron Bianchini (412) 268-7105 bianchini@ece.cmu.edu
PI Institution: Carnegie Mellon University
Contract title: Ultra-Dependable Real-Time Computing
Contract Number: N0001491J1304 N173
Reporting Period: 1 Oct 90 - 30 Sep 91

4 Transitions and DoD Interactions

Jay Strosnider interacts frequently with NOSC San Diego Distributed Combat Control project transitioning technology into Navy lab testbeds in San Diego. He also interacts with IBM, Bellcore and Intel on commercial applications of the developed technologies.

Ron Bianchini has interacted with the Naval Research Lab in Washington, D.C. concerning the internetworking of laboratory computing facilities. His commercial interactions include IBM Austin, AT&T Bell Labs and Bell Northern Research.

Principal Investigator Names:
Jay K. Strosnider (412) 268-6927 strosnider@ece.cmu.edu
Ron Bianchini (412) 268-7105 bianchini@ece.cmu.edu
PI Institution: Carnegie Mellon University
Contract title: Ultra-Dependable Real-Time Computing
Contract Number: N0001491J1304 N173
Reporting Period: 1 Oct 90 - 30 Sep 91

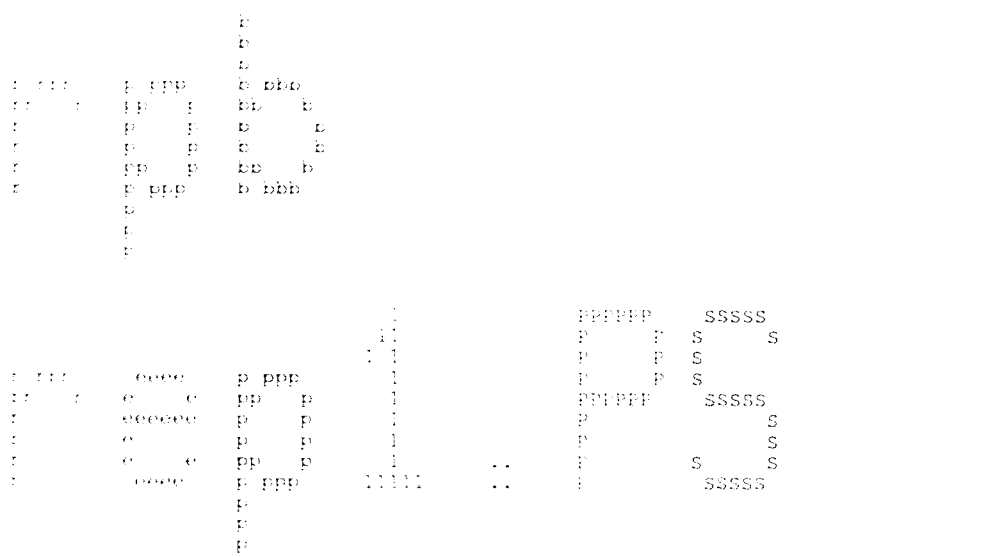
5 Software and Hardware Prototypes

The following prototype was developed partially under the sponsorship of the Office of Naval Research.

5.1 FAA Collision Avoidance Prototype

An initial distributed FAA collision avoidance Prototype was developed this year which simulates aircraft and in which some of the planes can contain integrated real-time artificial intelligence capabilities that control their behaviors. The first part of this project integrated the CROPS5 real-time production system, a lightweight process package, a remote procedure call package, the X window system, and the Unix timer facility. These components were integrated to create a coincidently real-time, distributed collision avoidance demonstration system running under Ultrix. Under this system, there is a single server process which maintains the information about the objects in the simulation. There are some client processes which generate "unintelligent" planes, which are planes on random, straight trajectories. The simulation may also contain "player" processes, in which a human being controls a plane from his workstation and can watch the progress of the other planes. The other clients are the "intelligent" planes which are under the control of a production system written in CROPS5. The production system currently detects when planes enter the "red zone" around its plane and attempts to change the course of the plane to avoid hitting any other planes. We plan to extend this distributed prototype to investigate and validate the dependability gains possible using integrated AI techniques.

Table of Contents



Job: repl.PS
Date: Mon Nov 4 10:24:29 1991